Resume Mining Frequent Patterns without Candidate Generation

Pola-pola yang sering muncul dalam data dalam transaksi database sering diteliti dalam data mining, dari penelitian sebelumnya mengadopsi apriori  dan pendekatannya, namun generasi yang diihasilkan masih membutuhkan biaya yang mahal, terutama terdapat pola proli dan/atau pola lama. Dalam studi ini penulis mengusulkan menggunakan FP-Tree Struktur, Gambaran Ide dari FP-Tree adalah mengkompres data yang besar menjadi struktur yang lebih ringkas, mengembangkan metode efisien pada FP-tree-based frequent pattern mining. Keluaran yang di inginkan adalah data yang padat tapi lengkap untuk menghindari biaya yang mahal dan perulangan dalam penscanan database, penambangan atau penemuan tugas menjadi lebih kecil, hanya menguji sub-basis data saja

FP Growth adalah metode untuk mengembangkan efisiensi FP Tree dengan tujuan untuk me-mining semua set dari FP menggunakan pattern fragment growth. Efisiensi mining bisa dilakukan menggunakan 3 teknik:

1. Kompresi database menjadi lebih kecil, sehingga struktur data juga jauh lebih kecil, diulang setiap database, dan menghemat biaya.
2. FP Tree mengadopsi metode pertumbuhan pola fragmen untuk menghemat biaya dari calon dataset yang besar.
3. Metode membagi dan menggabungkan dengan tujuan untuk menguraikan proses mining ke suatu set proses yang lebih kecil.

Proses FP Growth ini lebih cepat pada saat me-mining data dibanding dengan Algoritma Apriori, dan pertumbuhan marjin lebih luas ketika pola yang sering digunakan tumbuh lebih panjang. FP Growth juga melebihi algoritma Tree Projection. Metode mining berbasis FP Tree ini juga telah diuji pada database transaksi yang besar dalam aplikasi industri.

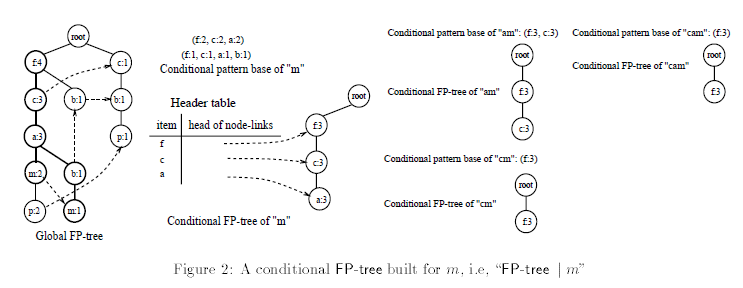
**FP Tree**

Langkah-langkah merancang struktur data yang padat:

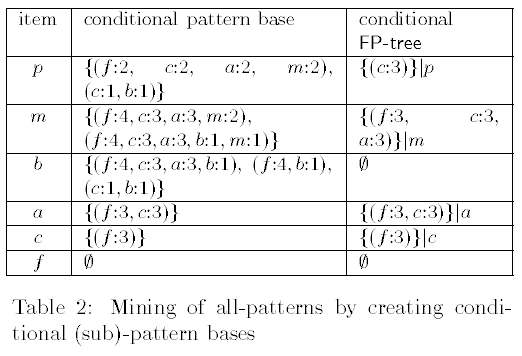
* Memindai database untuk mengidentifikasi set item yang sering terjadi.
* Jika pada saat menyimpan set item yang sering terjadi pada setiap transaksi di struktur yang padat, itu bisa menghindari pemindaian database yang berulangkali.
* Jika beberapa transaksi ada item set yang sama, bisa digabungkan menjadi satu dengan jumlah kejadian terdaftar sebagai count.
* Jika item yang sering diurutkan dalam urutan menurun, ada kemungkinan yang lebih baik untuk  membagi string pada transaksi yang lain.

**Mining Frequent Pattern Using FP Tree**

Hasil Conditional FP Tree

****

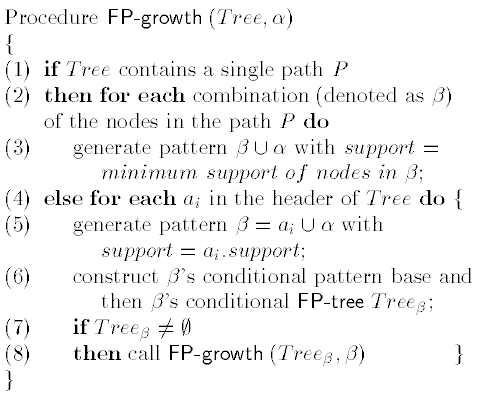
Untuk data m bisa dilihat di bawah ini:

****

**FP Growth: Mining Frequent Patterns with FP Tree by Pattern Fragment Growth**

Algoritma ini menghasilkan sekumpulan yang utuh dari frequent patterns. Dengan inputan FP Tree yang telah dikonstruksi yang menggunakan database dan minimum support treshold.

Langkah-langkah dari algoritma ini sebagai berikut:



Kesimpulan dan saran

*Frequent Pattern Tree* (FP-tree) merupakan struktur data baru yang berisi informasi-informasi penting dari pola data yang sering muncul. Informasi-informasi ini disimpan dalam bentuk kompresi. Sedangkan *Frequent Pattern Growth* (FP-growth) adalah metode untuk menggali pola data yang sering muncul pada database yang besar menggunakan FP-tree dan metode *divide and conquer.* Keunggulan menggunakan FP-growth adalah  sebagai berikut:

(1) Lebih cepat dan mudah dalam penggalian data karena Informasi dari database yang besar dimampatkan menjadi FP-tree.

(2) Lebih murah daripada algoritma apriori karena hanya menggunakan *pattern fragment growth*.

(3) Menggunakan divide and conquer yang mengurangi jumlah syarat pola berikutnya (*subsequent conditional pattern*) dan syarat FP-tree.

Untuk meningkatkan performa FP-growth hal-hal yang dapat dilakukan adalah sebagai berikut:

(1) Membagi database ke dalam beberapa bagian dan buat FP-tree untuk masing-masing bagian DB.

(2) Simpan FP-tree dalam *harddisk* menggunakan struktur tree B+ untuk mengurangi biaya I/O.

(3) Gunakan *minimum support threshold* yang kecil untuk mendapatkan hasil yang memuaskan.

(4) Jika terdapat data baru yang masuk dalam database, buat ulang FP-tree.